The true Processing In Memory accelerator
Key points

Unprecedented scalable ultra-efficient PIM* architecture and chip
- 4 Gb DRAM memory chips, embedding 8 processors on die
- Delivered as standard DDR4 2400 DIMM modules with 16 chips
- Server CPU helped by thousands of additional cores
- Boosting 20x data-intensive applications
- Power efficiency 10x better
  - By reducing drastically CPU-DRAM data movement
- At marginal cost

*Processing In Memory
Processing In Memory

• Put processors INSIDE the **main memory die**
  • Tackling dominant energy cost of data movement
• First implementation to meet success conditions
  • Up-to-date unmodified DRAM process
  • Mainstream memory interface & language support
• PIM more relevant than ever
  • More data intensive applications
  • Memory wall & end of Moore’s law
• Big data players
  • Computing efficiency now critical
  • Have scale & skills to adapt algorithms & SW

**Take away**

DRAM PIM tackles the dominant energy cost of data movement

PIM benefits more relevant than ever
  • New workloads
  • New players
UPMEM PIM-DRAM big data accelerator

- UPMEM DIMMs
  - Replacing standard DIMMs
  - DDR4 R-DIMM modules
    - 8GB+128DPUs (16 PIM chips)
- UPMEM PIM-DRAM chips
  - 4Gb DDR4 2400 DRAM + 8 DPUs @500MHz
  - Single die, standard 2x nm DRAM process
- Massive additional compute & bandwidth
  - 2TB/s DRAM-DPU BW for 128GB+2048 DPUs config
- Easily programmable SDK: C-programmable

Take away
Scalable as compatible with
- Current servers
- Unmodified DRAM process
- Programmers ;)
Samples & apps available

Copyright UPMEM® 2019
Standard DRAM package & DIMM

4Gb DRAM DDR4 2400 + 8DPUs @ 500MHz
1GB/s DRAM-DPU bandwidth
Standard DRAM package
~1cm² die – ~1,2W

Copyright UPMEM® 2019

DDR4 2400 R-DIMM module
UPMEM PIM massive benefits

- Massive speed-up
  - Massive additional compute & bandwidth
- Massive energy gains
  - Most data movement on chip
- Low cost
  - ~300$ of additional DRAM silicon
  - Affordable programming
- Massive ROI / TCO gains

<table>
<thead>
<tr>
<th>Energy efficiency when computing on or off memory chip</th>
<th>Server + PIM DRAM</th>
<th>Server + normal DRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM to processor 64-bit operand</td>
<td>pJ ~150</td>
<td>~3000*</td>
</tr>
<tr>
<td>Operation</td>
<td>pJ ~20</td>
<td>~10*</td>
</tr>
<tr>
<td>Server consumption</td>
<td>W ~700W</td>
<td>~300W</td>
</tr>
<tr>
<td>speed-up</td>
<td>~ x20</td>
<td>x1</td>
</tr>
<tr>
<td>energy gain</td>
<td>~ x10</td>
<td>x1</td>
</tr>
<tr>
<td>TCO gain</td>
<td>~ x10</td>
<td>x1</td>
</tr>
</tbody>
</table>


Copyright UPMEM® 2019
### Server with thousands of DPUs at work

<table>
<thead>
<tr>
<th>Field</th>
<th>Application</th>
<th>Benefits of PIM</th>
<th>Speed-up and TCO gain compared to same x86 server with standard DRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern matching</td>
<td>Genomics</td>
<td>Speed up comparison with reference data</td>
<td><strong>x25</strong> faster, evaluated by INRIA for DNA mapping* **</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>x41</strong> for NextGenMap with <strong>TCO 26x lower</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Speed up difference detections</td>
<td><strong>x25</strong> evaluated by UPMEM/INRIA for Illumina : DNA variant calling*, <strong>TCO 20 times better</strong>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Full mapping + variance analysis</td>
<td><strong>x22</strong> evaluated by UPMEM/INRIA (34’ vs 30h)****</td>
</tr>
<tr>
<td>Index DB</td>
<td>Index Search</td>
<td>Speed up queries &amp; latency</td>
<td><strong>x18</strong> speed-up - throughput, 1/100th latency</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>x14</strong> TCO gain</td>
</tr>
<tr>
<td>Analytics</td>
<td>Skyline multi-criteria analysis</td>
<td>More throughput efficient, easily scalable</td>
<td><strong>14x</strong> higher throughput evaluated by UCR****</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>10x</strong> better energy consumption</td>
</tr>
</tbody>
</table>

* Compared on Intel server with/without PIM on DRAM: simulations (generally 2048 DPUs/128GBs)
** 5 times better than GPU ; https://hal.archives-ouvertes.fr/hal-01327511/document ; https://ieeexplore.ieee.org/document/7822732 ; https://hal.archives-ouvertes.fr/hal-01294345/file/RR-BLAST_UPMEM_27_04_2016.pdf
*** Could vary with DPU pricing   **** Better efficiency than most advanced FPGA implementations; 30h is GATK

Copyright UPMEM® 2019
Multiple profiles for accelerated apps

No need to saturate bandwidths (DRAM or orchestration) nor minimize calculation

- DNA mapping-Pattern matching
- DNA Variant calling - Decompress & compare
- DNA Blast - Search and extend
- Decode & parse DB index
- Skyline - Multi-criteria sort
- Pathlen - Sparse matrix multiplication
- Dijkstra local - Graph search
- MD5 - Hash

Graph showing various metrics such as:
1. TCO gain (0 to 25)
2. Acceleration gain (0 to 45)
3. # DPUs used (0 to 4096)
4. # DPU cycles per input Byte (0 to 19000)
5. # threads used (0 to 16)
8. Orchestration time ratio (0 to 100%)
10. x86<->PIM system bandwidth consumed (0 to 110 GB/s)
11. Total DRAM bandwidth used within PIM system (0 to 2080 GB/s)
12. DPU bandwidth effective usage (0 to 100%)
The Hurdles on the road to the Graal

- DRAM process highly constrained
  - 3x slower transistors than same node digital process
  - Logic 10 times less dense vs. ASIC process
  - Routing density dramatically lower
    - 3 metals only for routing (vs. 10+), pitch x4 larger
- Strong design choices mandatory

But the PIM Graal is worth it!

Take away

DRAM vs. ASIC
- Far less performing
- Wafers 2x cheaper vs. ASIC

Leapfrogging Moore’s law
- Total Energy efficiency x10
- Massive, scalable parallelism
- Very low cost
Building a logic flow on a DRAM process

- Digital library & implementation flow created
- 4 different SRAM cuts created
  - 320 bits to 16 KB
  - Single port and dual ports
- DRAM IP
  - Modification to be minimized
  - The asynchronous interface increases the logic complexity

**Take away**

ASIC-like framework
- Logic cell library
- SRAM IPs
- Logic design & validation flow

Minimal DRAM IP modification
- DPU “added” to an otherwise mostly unmodified DRAM chip

Copyright UPMEM® 2019
Building a fast processor using slow transistors

- 14 pipe stages needed to reach 500 MHz
- Interleaved pipeline
  - No operand bypass, no stall signals
- 24 hardware threads
  - 100% performance achieved when 11 threads or more are running

**Take away**

DPU
- 1 instruction / cycle on multi-threaded code
- 1 GB/s from DRAM
- 8B to 2KB transfer

Equivalent to 1/6th of Xeon core on PIM applications (branchy, integer only code)

PIM server = 2048 DPUs
Multithreading allows a long pipeline to remain efficient

- DISPATCH . . . . . . . . Thread selection
- FETCH1/2/3 . . . . . . Instruction fetch
- READOP1/2/3 . . . . . register file access
- FORMAT . . . . . . . . operand formatting
- ALU1/2/3/4 . . . . . . Operators & WRAM access
- MERGE1/2 . . . . . . result formatting

Address & write data
Read data
Heavy multi-threading implies explicit memory hierarchy

- No data cache, 64 KB SRAM (WRAM) instead
  - Too much threading for caches
- No instruction cache, 24 KB SRAM (IRAM) instead
- DMA instructions move data between DRAM and WRAM/IRAM
  - Executed by an autonomous DMA engine, no/little effect on pipeline performance

Take away
Tightly coupled memories instead of caches:
- Too many threads for cache
- With efficient DMA instructions
An ISA optimized for the implementation styles that are realistic on DRAM process

Specific 32-bit ISA

• Aiming only scalar/in order/multithread implementation
• Providing efficient thread context
• Clean target of LLVM/CLANG
  • Regular triadic ISA
• Allowing out of the box compilation of 64-bit C code
  • Some 64-bit instructions
  • Helpers for 64-bit compilation

Take away

ARM® or RISC-V® discarded

Optimized ISA according to context
• Multithreaded, scalar, in order

Publicly documented ISA

Copyright UPMEM® 2019
A powerful ISA despite DRAM limitation

Beside supporting only 8x8 single cycle multiplies, DPU ISA more powerful than other 32-bit ISA.

• 0 cycle conditional jump on result properties
  • With rich set of jump conditions
• SHIFT+ADD/SUB instructions
• Rich set of logic instructions
  • Including NAND, NOR, ORN, ANDN, NXOR
• Rich set of shift/rotate instructions
• Large immediate values supported

Take away

ISA provides performance despite DRAM process
Clean sheet ISA approach helped significantly
Compatibility was not necessary

- DPU have no OS, neither need one
  - So many DPUs, no need to share one
- CLANG/LLVM tools are mature
- Explicit memory hierarchy mandatory
  - Would be an incompatibility point anyway
- Security is on our roadmap
  - No DPU sharing: dramatic security simplification
    - No side channel ever, by definition

Take away

No compatibility requirement
- No OS, no legacy binary
- CLANG/LLVM is the great enabler
- No need to ever share a DPU
  - Great security perspectives
Light server orchestration of DPUs

• DPU control registers mapped in physical memory space
  • Mapped in cacheable space

• Orchestration done through a software library, solving/hiding DDR4 related complexities
  • Bus width mismatch
  • Address interleaving
  • Lack of cache coherency
  • Lack of hardware arbitration

• Experience shows orchestration overhead is in the DPUs execution shadow

Take away

DDR4 not PIM friendly, but still OK
• Overhead dwarfed by DPU local calculations
• Complexity hidden in a programmer friendly library
The library feeds DPUs with correct data

Eight 64-bit “horizontal” words are turned into 8 vertical words, feeding 8 different DRAM chips.

This way DPUs see full 64-bit words, not chunk of them.

The library feeds DPUs with correct data.

Eight 64-bit “horizontal” words are turned into 8 vertical words, feeding 8 different DRAM chips.

This way DPUs see full 64-bit words, not chunk of them.

The transformation, a 8x8 matrix transposition, is done by the library inside a 64-byte cache line, thus very efficiently.
Programming thousands of cores

• Performance critical part of the application code moved to DPUs
  • Libraries helping for most common cases provided
• Server processors (x86, ARM64, POWER 9) acting as orchestrator
  • Still Executing the large majority of the application code (since non-performance critical)
  • Dispatching calculation intensive tasks to the DPUs
  • Collecting results from the DPUs
• Need to tackle data locality and compute parallelism
  • Largely experimented with labs and app owners

Take away
Server CPU act as orchestrator
• Application modifications limited to most intensive calculations
• Algorithm modification may be needed to exhibit higher % of local calculations
SDK at a glance
Samples ship Q3 2019, PIM FPGA & SW simulators available

- Chip sampled Q2 2019
  - Shipping from October
- SW simulator
  - SDK, doc & demo
  - Cloud9 graphical interface
  - Manage from personal user account
- Or FPGA fast app simulator
  - AWS f1.16x large instance
  - 256 DPUs @200MHz
- Both simulators available on AWS or on-premise
PIM, for real!
PoCs on verticals, samples, open sales start Q4 2019

• Production

• Go-to-Market

Visit upmem.com

Copyright UPMEM® 2019
Thank you!
Questions?